

Universidade de Aveiro, DETI

# Programação e Algoritmia

Guião das Aulas Práticas

Ano: 2022/2023

# Prática 1      Objetos e Classes

## Tópicos

- Classes e instâncias
- Construtores, atributos e métodos
- Getters e setters.
- Métodos especiais `__repr__` e `__str__`

Instale uma versão atualizada do Python seguindo as instruções disponibilizadas no Elearning<sup>1</sup>. Como referido nas instruções, na instalação, **ative a opção de atualizar variáveis de ambiente** (*environment variables*). Se tiver dúvidas ou dificuldades peça ajuda ao seu professor. Depois da instalação, deve testar a sua instalação como indicado nessas instruções.

Vai precisar também de um ambiente para desenvolvimento dos seus programas. Sugere-se a instalação do Visual Studio Code. Encontra informação sobre a sua instalação no Elearning.

## Exercício 1.1

Depois de garantidas as condições para conseguir programar em Python, complete a classe **Vehicle**, que tem como atributos das instâncias (objetos) `max_speed` e `kilometers`.

```
class Vehicle:
    def __init__(self, max_speed, kilometers):
    def set_max_speed(self, max_sp):
    def get_max_speed(self): # returns value of max_speed attribute
    def set_kilometers(self, kms):
    def get_kilometers(self): # returns value of kilometers attribute
```

Teste com:

```
modelX = Vehicle(240, 18)
print(modelX.max_speed, modelX.kilometers)
modelX.set_kilometers(1000)
modelX.set_max_speed(200)
print(f"Vehicle has {modelX.kilometers} Km and a max speed of {modelX.max_speed} Kms/h")
```

## Exercício 1.2

Complete a classe para números complexos abaixo implementando os métodos **add()** e **subtract()**. Implemente também o método especial **\_\_str\_\_()**.

```
class Complex:
    """Class represents a complex number."""
```

---

<sup>1</sup> <https://elearning.ua.pt/mod/page/view.php?id=1222392>

```
def __init__(self, real, imaginary):
    """Initialize Complex class's attributes."""
    self.real = real
    self.imaginary = imaginary
def __repr__(self):
    """Return string representation"""
    return (f'({self.real}' +
            (' + ' if self.imaginary >= 0 else ' - ') +
            f'{abs(self.imaginary)}i)')
```

Teste com o seguinte código:

```
cmplx1= Complex(1,1)
cmplx2= Complex(2,2)
print(cmplx1)
print(cmplx2)
print(cmplx1.add(cmplx2))
print(cmplx1.subtract(cmplx2))
print(f"{cmplx1} + {cmplx2} = {cmplx1.add(cmplx2)}")
print(f"{cmplx2} + {cmplx1} = {cmplx2.add(cmplx1)}")
print(f"{cmplx2} - {cmplx1} = {cmplx2.subtract(cmplx1)}")
```

Crie um programa que efetue as duas operações com todas as combinações de um conjunto de números definido por si (guardados numa lista).

[TPC] Adicione o método **multiply()** à classe e adapte o seu código de teste para incluir esta nova operação.

### Exercício 1.3

Crie uma classe adequada à modelação de **um ponto** (definido pelas coordenadas reais x e y, em metros). Inclua um método **distancia()** para calcular distância entre 2 pontos.

Crie uma lista com 5 pontos, com coordenadas aleatórias. Calcule e imprima a distância entre todos os pares de pontos.

### Exercício 1.4

Implemente classes que permitam criar objetos das seguintes **formas geométricas**: Círculo, Quadrado e Retângulo.

Cada forma é caracterizada por dois atributos obrigatórios – cor (String) e centro (ponto com coordenadas x, y) – e ainda atributos adicionais para cada tipo de forma (raio para o círculo, dois lados no retângulo e um lado no caso do quadrado). Tenha em atenção os conceitos de encapsulamento. Reutilize a classe do exercício 1.3 para o centro.

Garanta as seguintes especificações:

- crie classes que representem cada uma das figuras geométricas, implementando construtor(es) adequados para cada classe;
- adicione todos os métodos getters e setters relevantes;
- implemente um método para calcular a área de cada tipo de figura;
- [TPC] implemente um método para calcular o perímetro de cada tipo de figura;
- [TPC] implemente um método que desenhe a forma geométrica usando o módulo Turtle;
- implemente um método para verificar se os dois círculos se intersectam;

Em paralelo com o desenvolvimento das suas classes vá desenvolvendo um programa que lhe permita testar todas as classes criadas e os seus métodos.

## Exercício 1.5

Pretende-se construir um sistema de informação simplificado para a gestão da biblioteca de uma universidade. A biblioteca contém um catálogo de livros e um conjunto de utilizadores (só alunos). Como primeiro passo pretende-se criar Classes para Livros e Alunos.

Considerando que:

- Todos os alunos são identificados pelo seu número mecanográfico, nome e curso;
- Os livros são caracterizados por um ID (numérico e sequencial, começando em 100), título e tipo de empréstimo (CONDICIONAL ou NORMAL).

Desenvolva as classes Livro e Aluno e teste-as com o programa seguinte:

```
def main():
    livro1= Livro("Java 8", "CONDICIONAL");
    print(livro1)

    catalogo=[]
    catalogo.append(livro1)
    catalogo.append(Livro("POO em Java 8"))
    catalogo.append(Livro("Java para totós", "NORMAL"))

    print(f"ID = {catalogo[1].getId()} Título = { catalogo[1].getTitulo()}")
    catalogo[2].setTipoEmprestimo("CONDICIONAL")

    for livro in catalogo:
        print(livro)

    # Alunos
    utilizadores=[]
    utilizadores.append(Aluno("Catarina Marques", 80232, "MIEGI"))
    utilizadores.append(Aluno("Joao Silva", 90123, "Lic Física"))
    utilizadores[1].set_num_mec(80123)
    utilizadores.append(Aluno("António Costa", 100123, "Lic Matemática"))

    for aluno in utilizadores:
        print(aluno)

main()
```

Cujo resultado da sua execução deve ser:

```
Livro 100: Java 8, CONDICIONAL
ID = 101 Título = POO em Java 8
Livro 100: Java 8, CONDICIONAL
Livro 101: POO em Java 8, NORMAL
Livro 102: Java para totós, CONDICIONAL
Aluno: 80232; Catarina Marques; MIEGI
Aluno: 80123; Joao Silva; Lic Física
Aluno: 100123; António Costa; Lic Matemática
```

## Exercício 1.6

Desenvolva uma classe Data tendo por atributos dia, mês e ano que lhe permita executar o seguinte programa. Pode fazer uso de código que desenvolveu em Introdução à Programação.

```
data1 = Data (28,2,2022)
print(data1)
data1.nextDay()
print (data1)
print(f"Month 2 of 2022 has {Data.monthDays(2, 2022)} days")
```